



CPSC203 – Introduction to Problem Solving and Using Application Software

Fall 2009

Tutorial 25, Mehrdad Nurolahzade

Introduction

- Problem Solving Review

Count Down (1)

- Write the function **count_down(n)** that given Integer argument **n** counts down and prints out numbers from **n** to 0.

- Example run:

```
count_down(3)
```

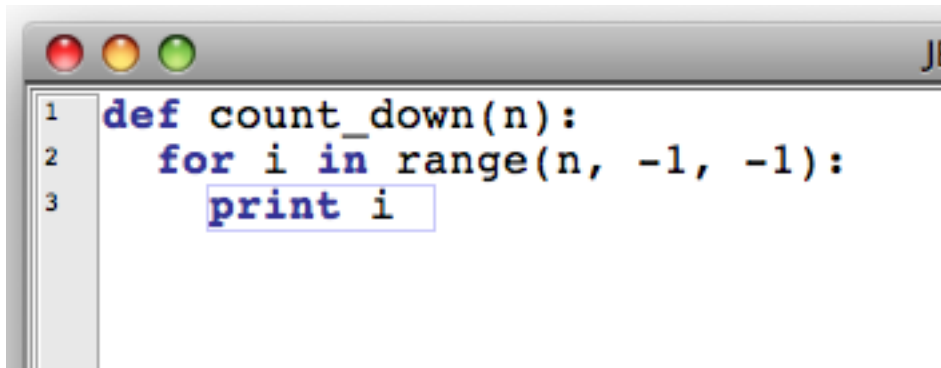
```
3
```

```
2
```

```
1
```

```
0
```

Count Down (2)

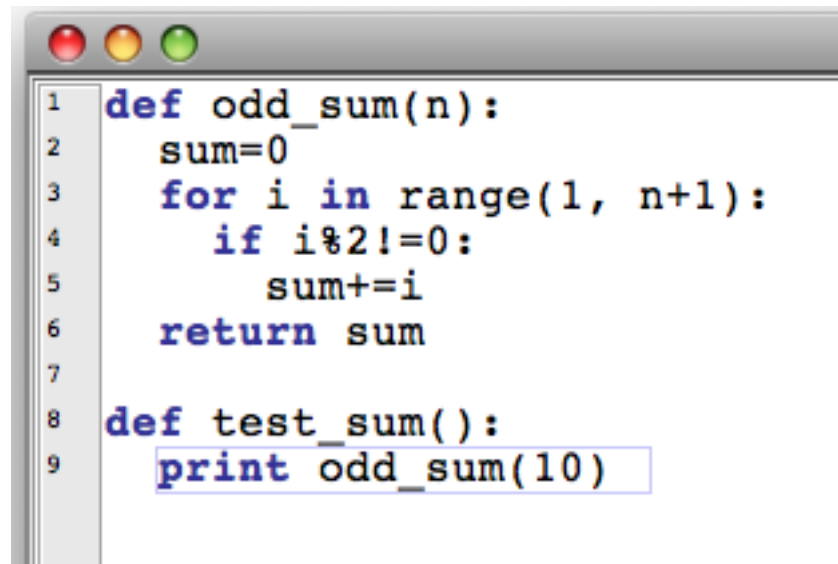


```
1 def count_down(n):  
2     for i in range(n, -1, -1):  
3         print i
```

Odd Sum (1)

- Write the function **odd_sum(n)** that given the argument **n** of type Integer returns the sum of odd numbers from 1 to **n**.
- Write this other function **test_sum()** that uses function **odd_sum()** to print out sum of odd numbers between 1 and 10.
- Example run:
`test_sum()`
25

Odd Sum (2)



```
1 def odd_sum(n):
2     sum=0
3     for i in range(1, n+1):
4         if i%2!=0:
5             sum+=i
6     return sum
7
8 def test_sum():
9     print odd_sum(10)
```

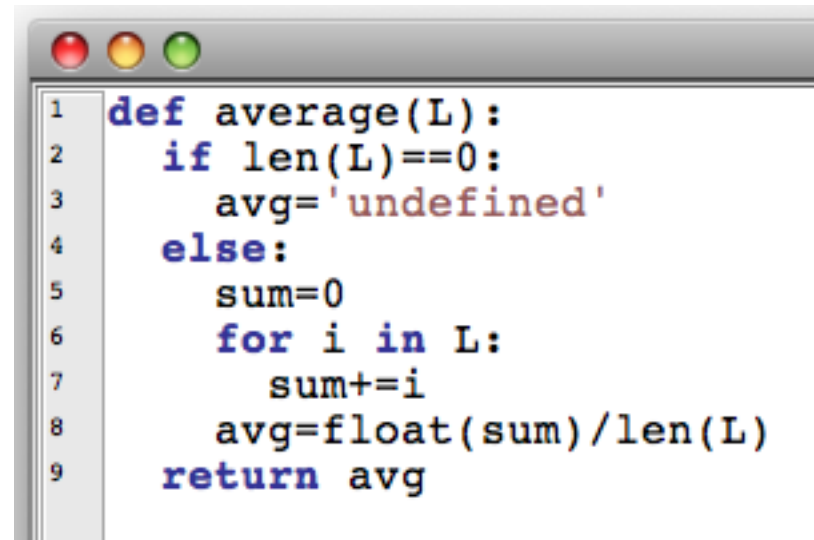
List Average (1)

- Write the function **average(L)** that given the argument **L** of type List returns the average of elements in the **L**.

- Example run:

```
average([1, 2, 2, 3, 5])  
2.6
```

List Average (2)



```
1 def average(L):  
2     if len(L)==0:  
3         avg='undefined'  
4     else:  
5         sum=0  
6         for i in L:  
7             sum+=i  
8         avg=float(sum)/len(L)  
9     return avg
```

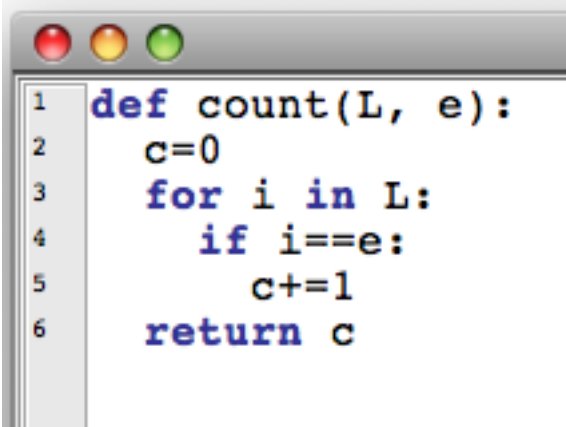

Count of Elements (1)

- Write the function **count(L, e)** that given the argument **L** of type list and element **e**, returns the count of elements **e** in the **L**.

- Example run:

```
count(['a', 'b', 'a', 'c', 'd'], 'a')  
2
```

Count of Elements (2)

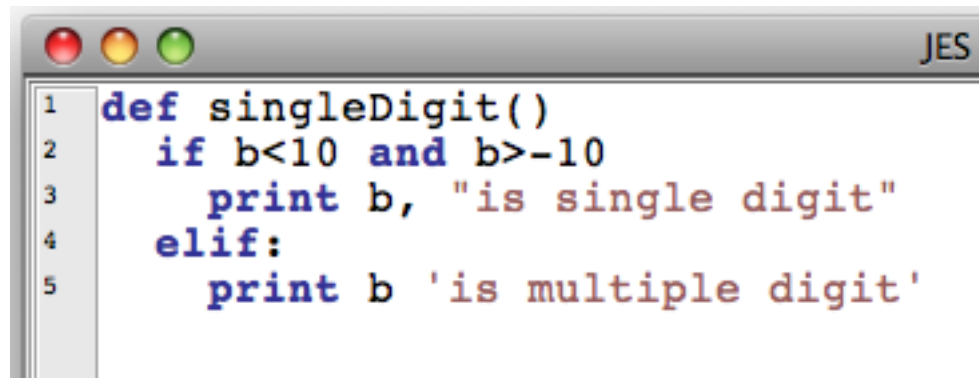
A screenshot of a code editor window with a standard macOS-style title bar (red, yellow, and green buttons). The editor contains a Python function definition. The code is as follows:

```
1 def count(L, e):  
2     c=0  
3     for i in L:  
4         if i==e:  
5             c+=1  
6     return c
```

Find Mistakes (1)

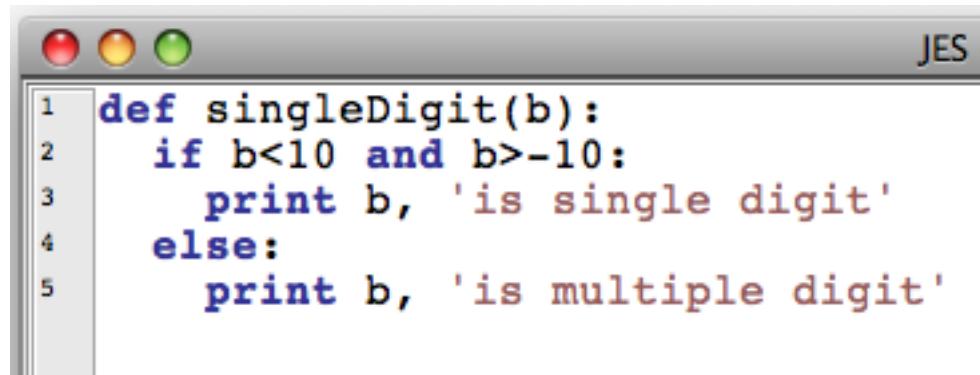
- The function **singleDigit(b)** which accepts an argument **b**, which is a number and prints out the message '... is single digit' or '... is multiple digit' if **b** is single or multiple digit respectively.
- However, the given function contains 5 mistakes. Correct these mistakes.

Find Mistakes (2)

A screenshot of a JES (Jupyter Environment Shell) window. The window has a title bar with three colored buttons (red, yellow, green) and the text 'JES' on the right. The code is displayed in a monospaced font with syntax highlighting. The code is as follows:

```
1 def singleDigit()  
2     if b<10 and b>-10  
3         print b, "is single digit"  
4     elif:  
5         print b 'is multiple digit'
```

Find Mistakes (3)



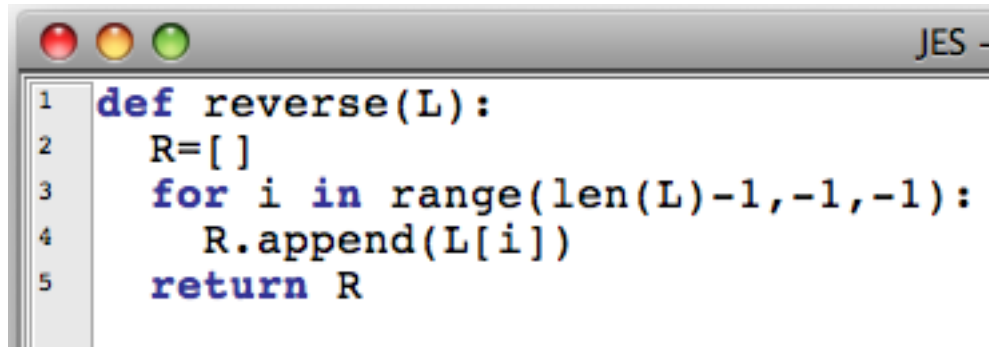
```
1 def singleDigit(b):
2     if b<10 and b>-10:
3         print b, 'is single digit'
4     else:
5         print b, 'is multiple digit'
```

Reverse (1)

- Write function **reverse(L)** that accepts an argument **L** of type list and return a list with the same elements in reverse order.
- Note: you are allowed to use **L.reverse()** function.
- Example run:

```
reverse([2, 3, 5, 2, 1, 6])  
[6, 1, 2, 5, 3, 2]
```

Reverse (2)

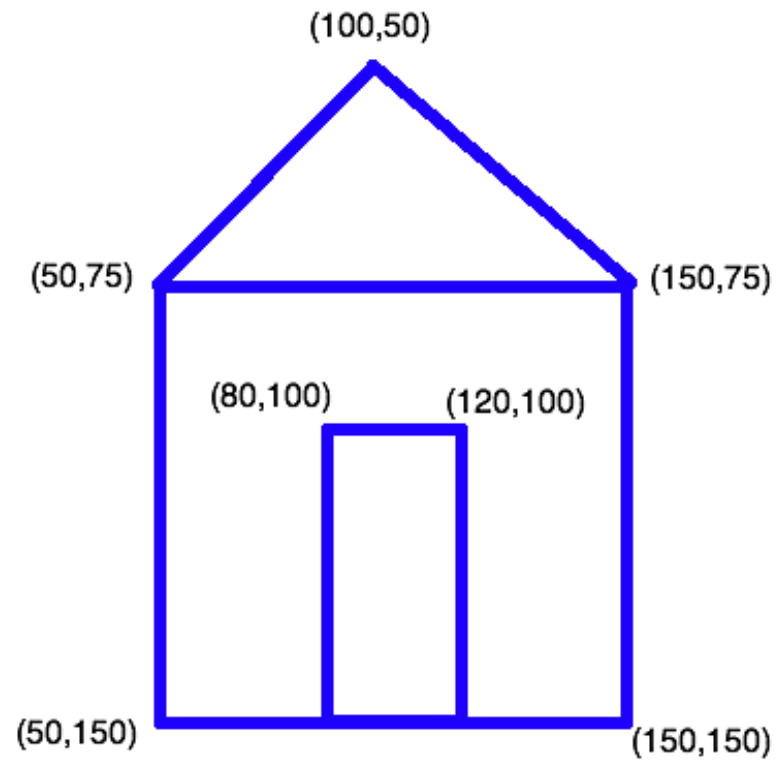
A screenshot of a JES (Jupyter Environment Shell) window. The window has a title bar with three colored buttons (red, yellow, green) and the text "JES -". Inside the window, a Python function named "reverse" is defined. The function takes a list "L" as an argument. It initializes an empty list "R". It then uses a "for" loop with "range(len(L)-1, -1, -1)" to iterate over the indices of "L" in reverse order. For each index "i", it appends "L[i]" to "R". Finally, it returns "R".

```
1 def reverse(L):  
2     R=[]  
3     for i in range(len(L)-1,-1,-1):  
4         R.append(L[i])  
5     return R
```

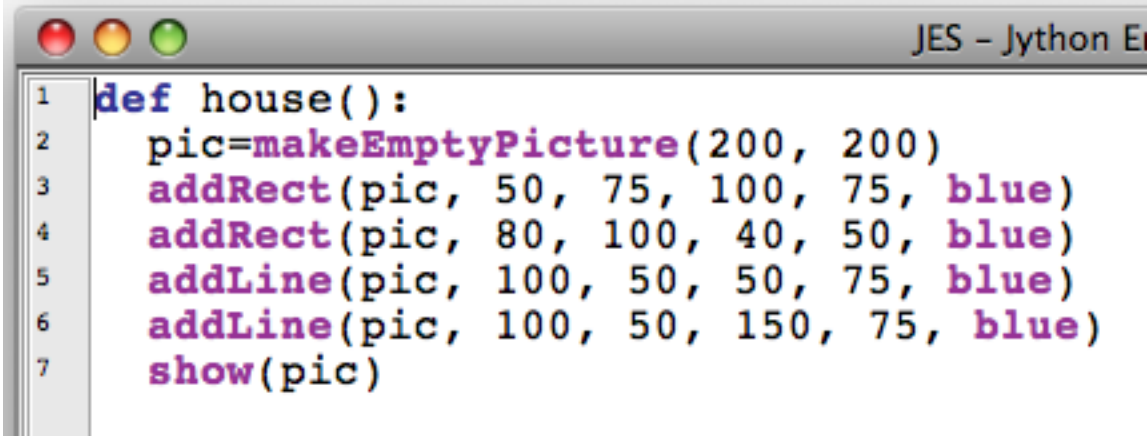
House (1)

- Write function **house()** that draws and shows the picture given at the next page. The provided coordinate are for clarification only and will not be part of the picture.

House (2)



House (3)



A screenshot of a Jython Editor (JES) window. The window has a title bar with three colored buttons (red, yellow, green) on the left and the text "JES - Jython Editor" on the right. The main area contains a Python function definition for drawing a house. The code is as follows:

```
1 def house():
2     pic=makeEmptyPicture(200, 200)
3     addRect(pic, 50, 75, 100, 75, blue)
4     addRect(pic, 80, 100, 40, 50, blue)
5     addLine(pic, 100, 50, 50, 75, blue)
6     addLine(pic, 100, 50, 150, 75, blue)
7     show(pic)
```

Prime Numbers (1)

- Function **isPrime(number)** checks if the input parameter **number** is a prime number or not. If it is, it returns "true", otherwise it returns "false".
- Write function **checkForPrimes(L)** which receives **L**, a list of numbers, and by calling function **isPrime(number)** for every element of the **L** determines whether that element is prime or not. It prints out "<number> is prime." for prime numbers and "<number> is not prime." for non-prime numbers.
- Example run:

```
checkForPrimes([1, 2, 4])  
1 is prime.  
2 is prime.  
4 is not prime.
```

Prime Numbers (2)

```
1 def isPrime(number):
2     divisors = 0
3     for i in range(2, number):
4         if number % i == 0:
5             divisors = divisors + 1
6     if divisors == 0:
7         return True
8     else:
9         return False
10
11 def checkForPrimes(L):
12     for i in L:
13         if isPrime(i):
14             print i, 'is Prime.'
15         else:
16             print i, 'is not prime.'
```